

Student name: Lê Ngọc Anh Vũ

Student ID: 20236014

Asignment 1: Create a new project to implement the Home Assigment 1. Compile and upload to simulator. Initialize two operands (register s1 and s2), run this program step by step, observe memory and registers value.

Case 1: s1 and s2 have different sign

We initialize s1=20232024, s2= -134

The screenshot shows the ELab 4 - RARS 1.6 interface. The assembly code window displays the following instructions:

```

    .Text Segment
    Bkpt Address Code Basic Source
    0x00400000 0x0134b4b7 lui x9,4939
    0x00400004 0x75848493 addi x9,x9,1880
    0x00400008 0x7fa00913 addi x10,x0,-134
    0x0040000c 0x00000293 addi x5,x0,0
    0x00400010 0x012489b3 add x19,x9,x18
    0x00400014 0x0124c333 xor x6,x9,x18
    0x00400018 0x00034e63 blt x6,x0,28
    0x0040001c 0x0099a3b3 slt x7,x19,x9
    0x00400020 0x0004c663 blt x9,x0,12

    .Data Segment
    Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
    0x10010000 0 0 0 0 0 0 0 0
    0x10010020 0 0 0 0 0 0 0 0
    0x10010040 0 0 0 0 0 0 0 0
    0x10010060 0 0 0 0 0 0 0 0
    0x10010080 0 0 0 0 0 0 0 0
    0x100100a0 0 0 0 0 0 0 0 0
    0x100100c0 0 0 0 0 0 0 0 0
  
```

The Registers window shows the following values:

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	20230144
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0

The Messages window shows the message: "program is finished running(dropped off bottom)".

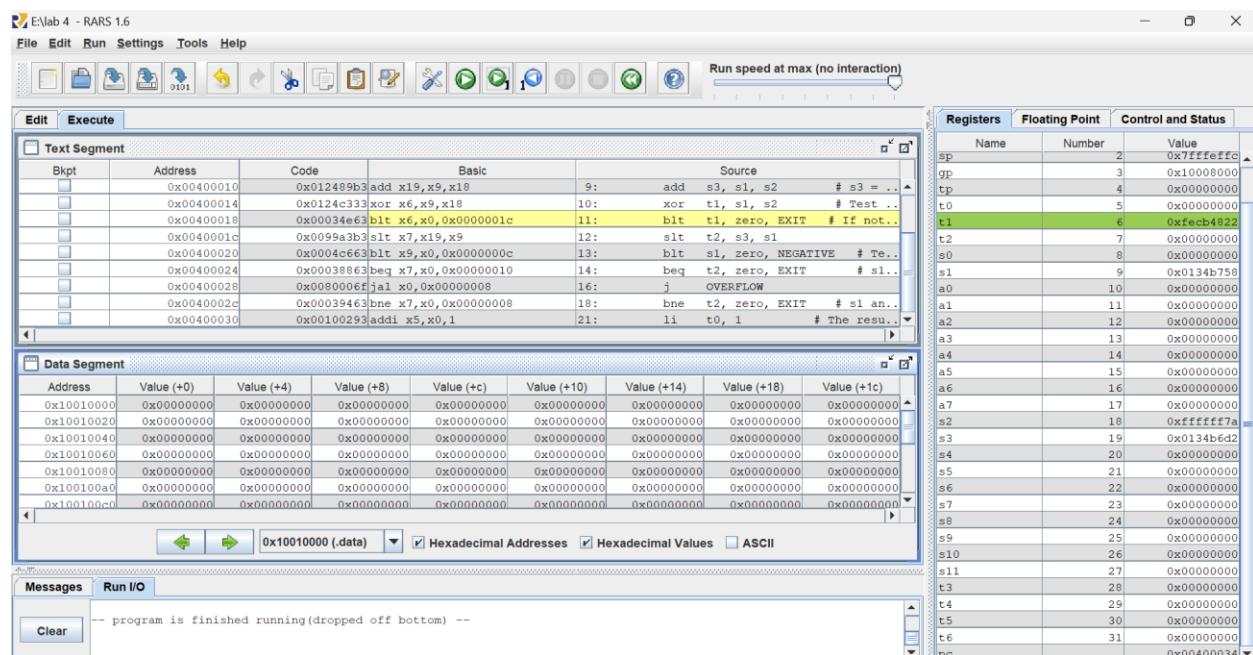
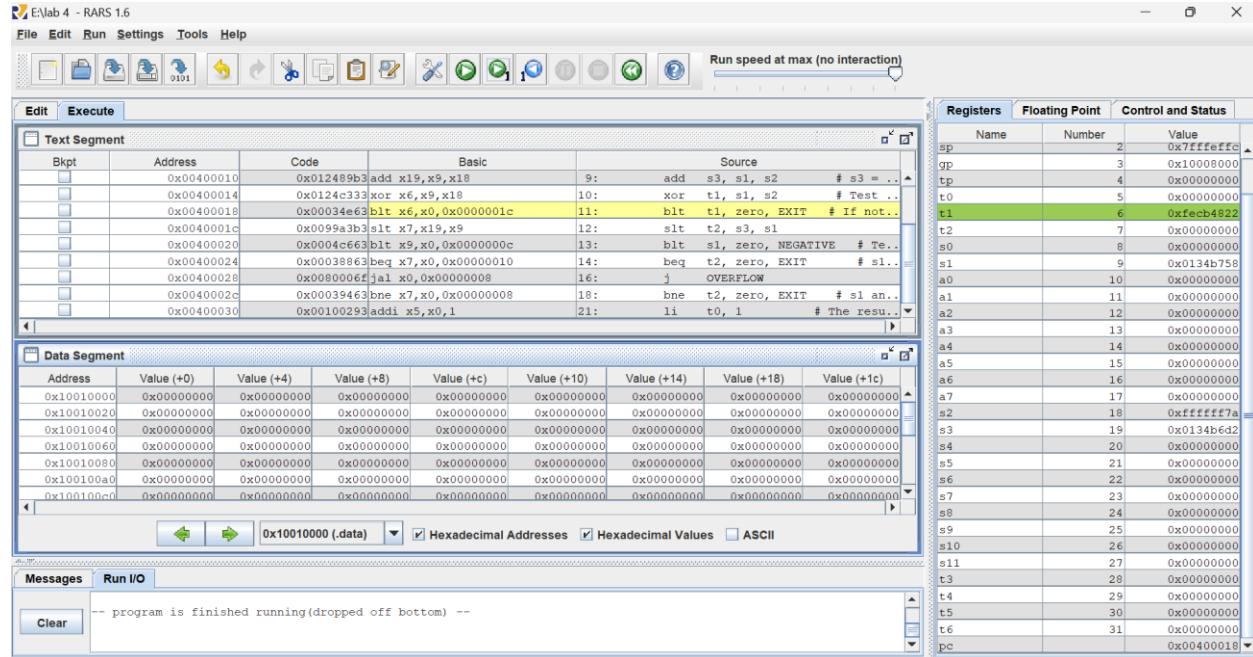
The screenshot shows the ELab 4 - RARS 1.6 interface. The assembly code window displays the same instructions as the previous screenshot. The Registers window shows the following values:

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	0
s1	9	20232024
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	-134
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc	4194316	

The Messages window shows the message: "program is finished running(dropped off bottom)".

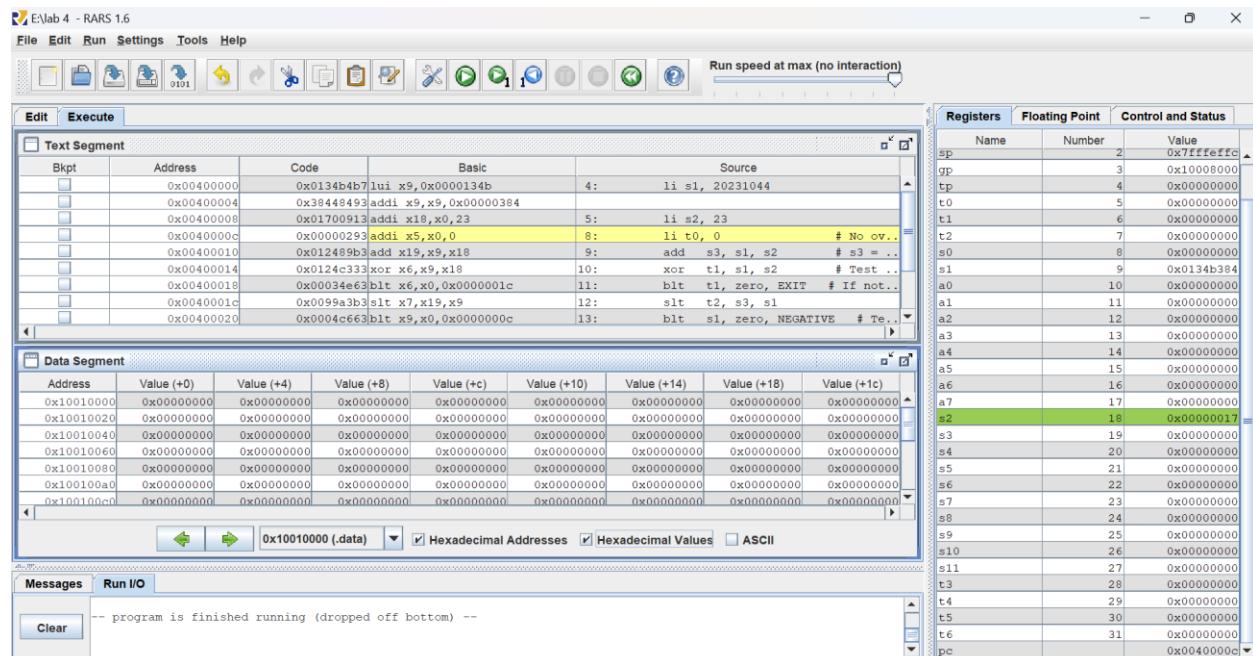
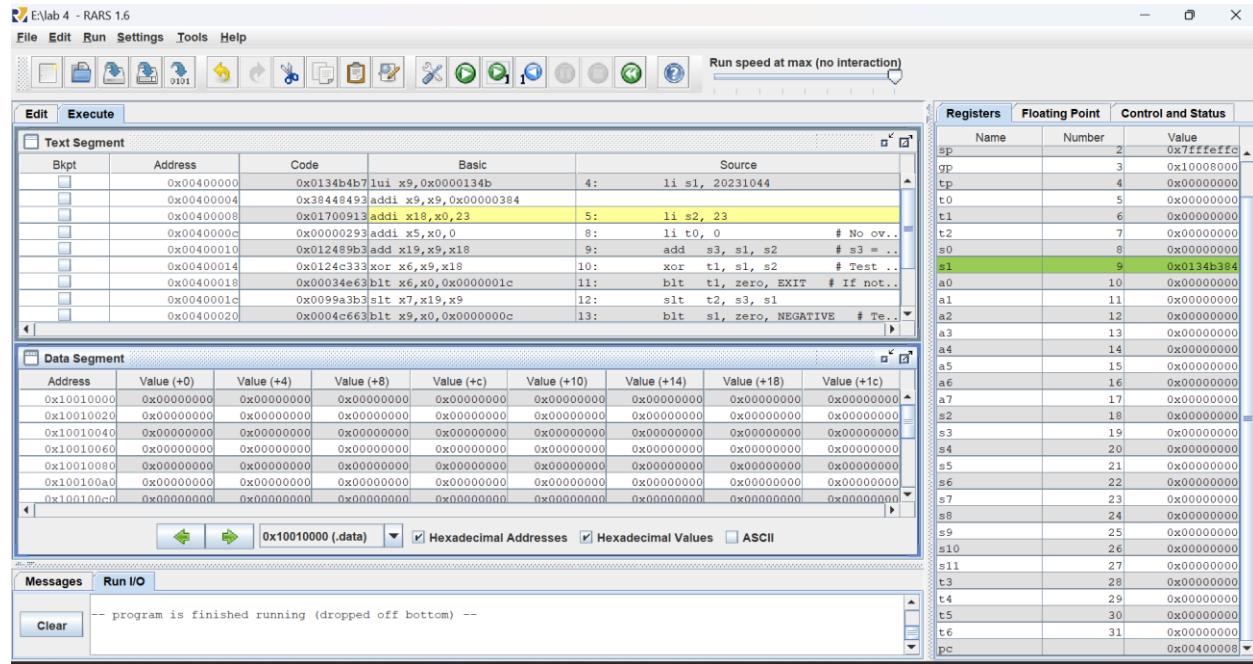
Because s1 and s2 have different sign, register t1 less than 0 and program jumps to EXIT label

The register pc jumps from address 0x00400018 to address 0x00400034 and then the program ends.



Case 2: s1 and s2 have the same signe but s1+s2 doesn't overflow.

We initialize s1=20231044 and s2=23



Because s1 and s2 have the same sign, the instruction "slt t2, s3, s1" is executed.

Because s3=s1+s2 is greater than s1, the program jumps to EXIT label and the register pc jumps from address 0x00400024 to 0x00400034

Then, the program ends.

ELab 4 - RARS 1.6

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Registers

Name	Number	Value
sP	2	0x7ffffeffc
gP	3	0x100000000
tP	4	0x000000000
t0	5	0x000000000
t1	6	0x0134b393
t2	7	0x00000000
s0	8	0x000000000
s1	9	0x0134b384
a0	10	0x000000000
a1	11	0x000000000
a2	12	0x000000000
a3	13	0x000000000
a4	14	0x000000000
a5	15	0x000000000
a6	16	0x000000000
a7	17	0x000000000
s2	18	0x0000000017
s3	19	0x0134b39b
s4	20	0x000000000
s5	21	0x000000000
s6	22	0x000000000
s7	23	0x000000000
s8	24	0x000000000
s9	25	0x000000000
s10	26	0x000000000
s11	27	0x000000000
t3	28	0x000000000
t4	29	0x000000000
t5	30	0x000000000
t6	31	0x000000000
pc	32	0x00400024

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400010	0x012489b3	add x19,x9,x18	9: add s3, sl, s2 # s3 = ..
	0x00400014	0x0124c333	xor x6,x9,x18	10: xor t1, sl, s2 # Test ..
	0x00400018	0x00034e63	blt x6,x0,0x0000001c	11: blt t1, zero, EXIT # If not..
	0x0040001c	0x0099a3b3	slt x7,x19,x9	12: slt t2, s3, sl
	0x00400020	0x0004c663	blt x9,x0,0x0000000c	13: blt sl, zero, NEGATIVE # Te..
	0x00400024	0x00038863	beq x7,x0,0x00000010	14: beq t2, zero, EXIT # s1 ..
	0x00400028	0x0080006f	jal x0,0x00000008	16: j OVERFLOW
	0x0040002c	0x00039463	bne x7,x0,0x00000008	18: bne t2, zero, EXIT # s1 an..
	0x00400030	0x00100293	addi x5,x0,1	21: li t0, 1 # The resu..

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages

-- program is finished running (dropped off bottom) --

Run I/O

Clear

EElab 4 - RARS 1.6

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Registers

Name	Number	Value
sP	2	0x7ffffeffc
gP	3	0x100000000
tP	4	0x000000000
t0	5	0x000000000
t1	6	0x0134b393
t2	7	0x00000000
s0	8	0x000000000
s1	9	0x0134b384
a0	10	0x000000000
a1	11	0x000000000
a2	12	0x000000000
a3	13	0x000000000
a4	14	0x000000000
a5	15	0x000000000
a6	16	0x000000000
a7	17	0x000000000
s2	18	0x0000000017
s3	19	0x0134b39b
s4	20	0x000000000
s5	21	0x000000000
s6	22	0x000000000
s7	23	0x000000000
s8	24	0x000000000
s9	25	0x000000000
s10	26	0x000000000
s11	27	0x000000000
t3	28	0x000000000
t4	29	0x000000000
t5	30	0x000000000
t6	31	0x000000000
pc	32	0x00400034

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400010	0x012489b3	add x19,x9,x18	9: add s3, sl, s2 # s3 = ..
	0x00400014	0x0124c333	xor x6,x9,x18	10: xor t1, sl, s2 # Test ..
	0x00400018	0x00034e63	blt x6,x0,0x0000001c	11: blt t1, zero, EXIT # If not..
	0x0040001c	0x0099a3b3	slt x7,x19,x9	12: slt t2, s3, sl
	0x00400020	0x0004c663	blt x9,x0,0x0000000c	13: blt sl, zero, NEGATIVE # Te..
	0x00400024	0x00038863	beq x7,x0,0x00000010	14: beq t2, zero, EXIT # s1 ..
	0x00400028	0x0080006f	jal x0,0x00000008	16: j OVERFLOW
	0x0040002c	0x00039463	bne x7,x0,0x00000008	18: bne t2, zero, EXIT # s1 an..
	0x00400030	0x00100293	addi x5,x0,1	21: li t0, 1 # The resu..

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages

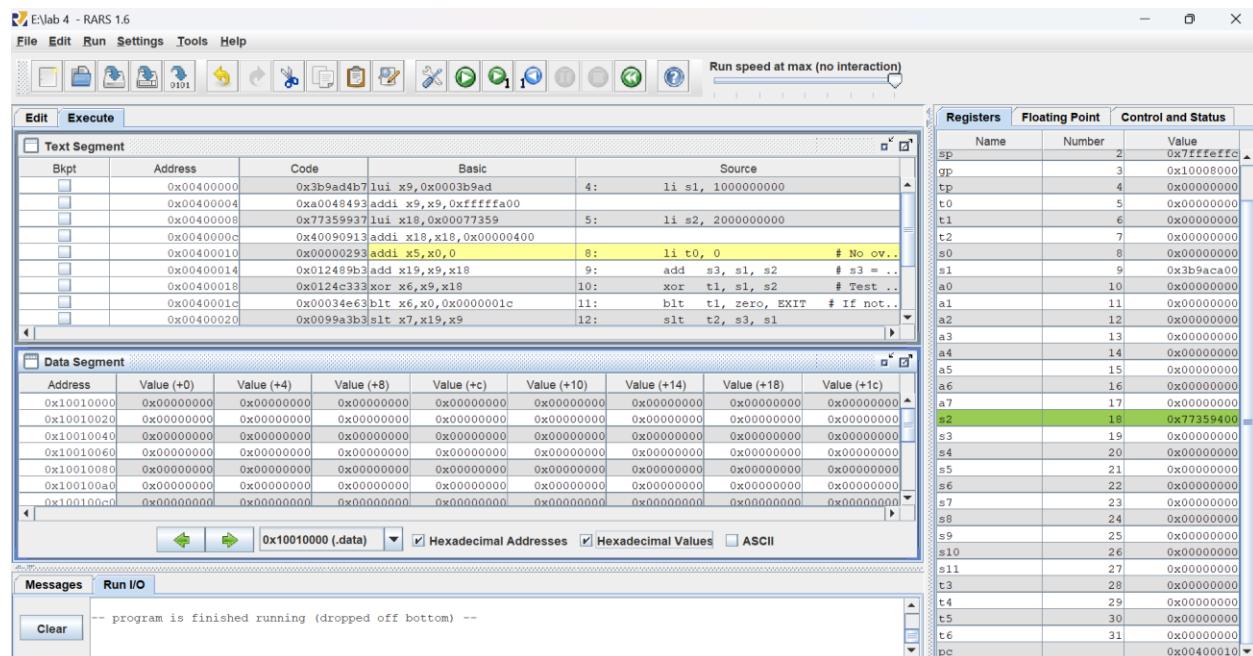
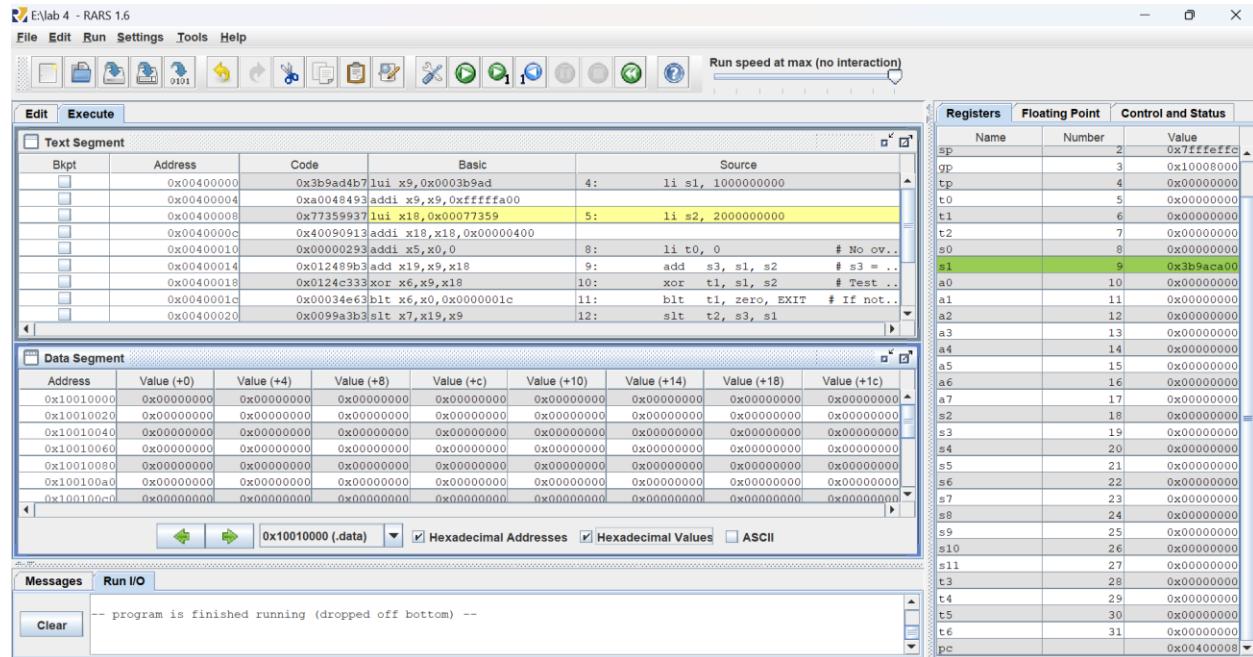
-- program is finished running (dropped off bottom) --

Run I/O

Clear

Case 3: s1 and s2 have the same sign and s3=s1+s2 overflows.

We initialize s1=1000000000, s2=2000000000



Because s1 and s2 have the same sign, the register t1>0 and the program don't jump to EXIT label.

The screenshot shows the ELab 4 - RARS 1.6 debugger interface. The assembly code window displays the following instructions:

```

Text Segment
Bkpt Address Code Basic Source
0x00400014 0x012489b3 add x19,x9,x18 9: add s3, s1, s2 # s3 = ...
0x00400018 0x0124c333 xor x6,x9,x18 10: xor t1, s1, s2 # Test ...
0x0040001c 0x00034e63 blt x6,x0,0x00000001c 11: blt t1, zero, EXIT # If not...
0x00400020 0x0099a3b3 silt x7,x19,x9 12: silt t2, s3, s1
0x00400024 0x0004c663 blt x9,x0,0x0000000c 13: blt s1, zero, NEGATIVE # Te...
0x00400028 0x00038863 beq x7,x0,0x00000010 14: beq t2, zero, EXIT # s1...
0x0040002c 0x0008006f jal x0,0x00000008 16: j OVERFLOW
0x00400030 0x00039463 bne x7,x0,0x00000008 18: bne t2, zero, EXIT # s1 an...
0x00400034 0x00100293 addi x5,x0,1 21: li t0, 1 # The resu...

```

The Registers window shows the following values:

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x4caf5e00
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x3b9aca00
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x77359400
s3	19	0xb2d05e00
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040001c

The Data Segment window shows memory starting at address 0x10010000. The Messages window indicates "program is finished running (dropped off bottom)".

We compare the register $s3=s1+s2$ with register $s1$.

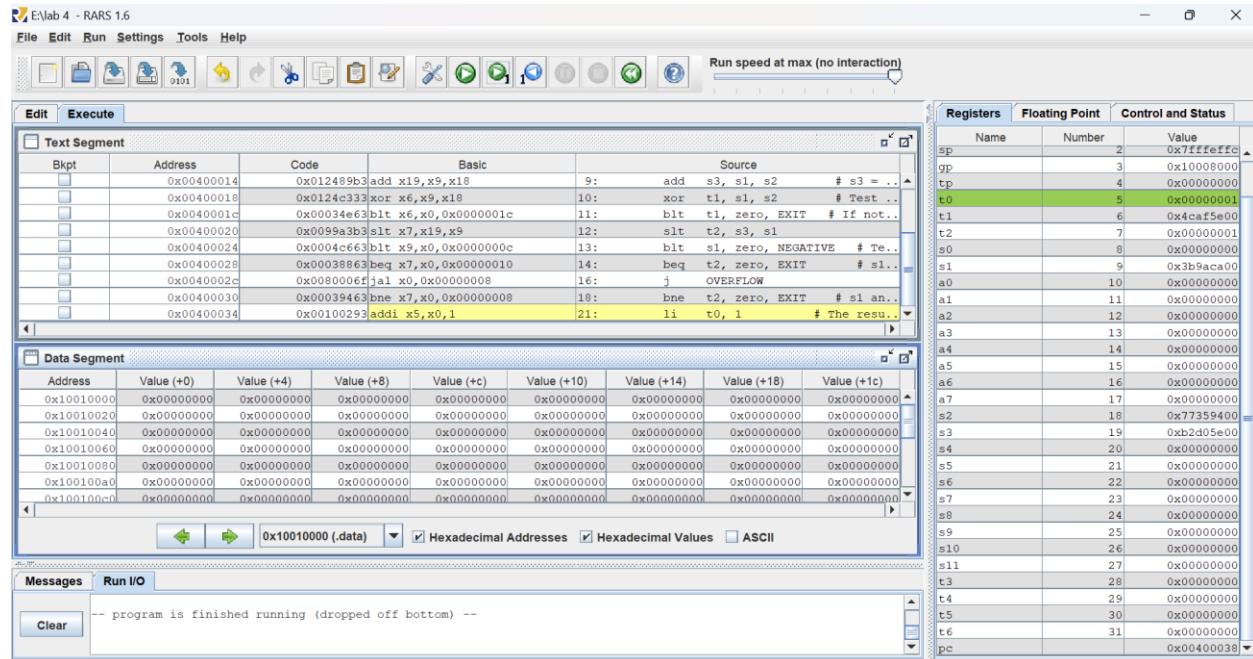
The screenshot shows the ELab 4 - RARS 1.6 debugger interface. The assembly code window displays the same set of instructions as the previous screenshot.

The Registers window shows the following values:

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x4caf5e00
t2	7	0x00000001
s0	8	0x00000000
s1	9	0x3b9aca00
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x77359400
s3	19	0xb2d05e00
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400024

The Data Segment window shows memory starting at address 0x10010000. The Messages window indicates "program is finished running (dropped off bottom)".

Because $s3 < s1$ and $s1, s2$ are positive, the register $t0$ gets value 1. It means that overflow occurred in the program.



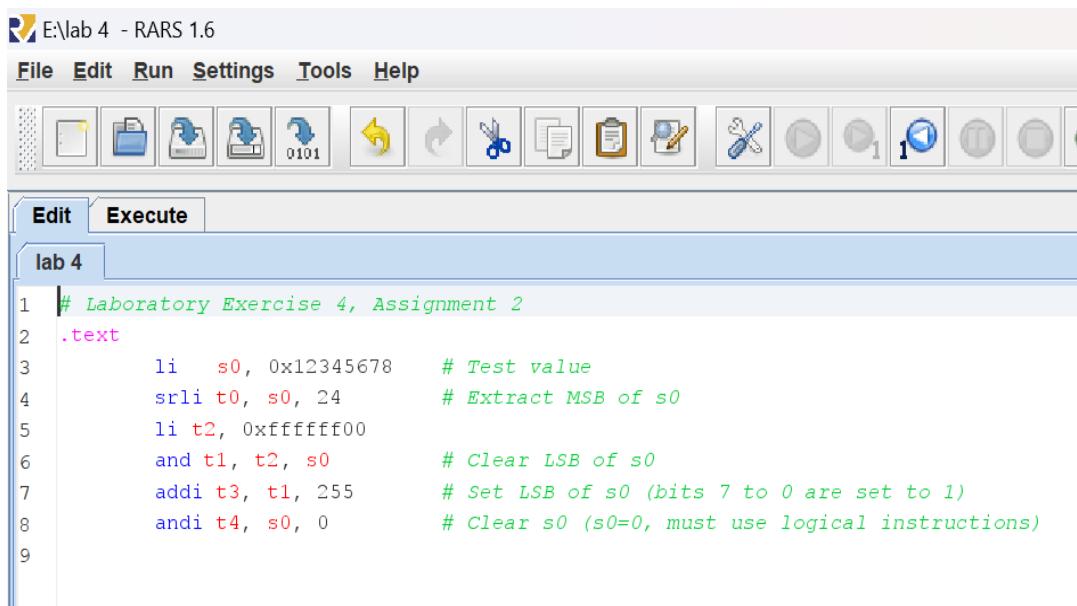
The program ends.

Assignment 2:

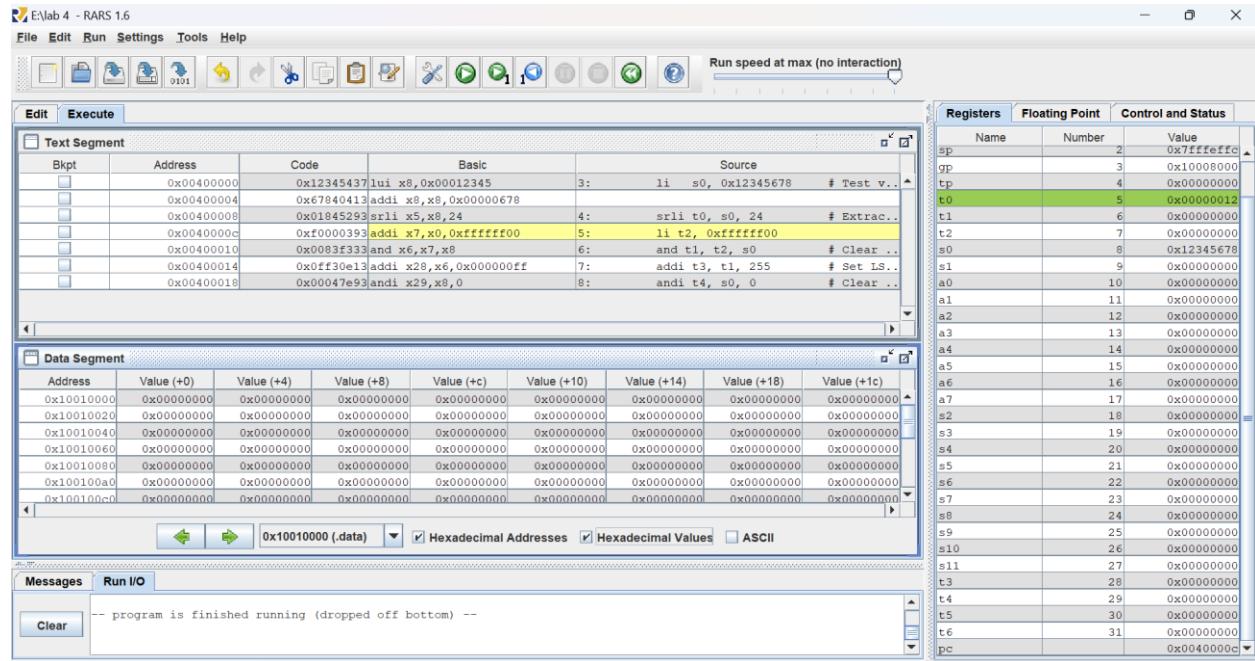
Write a program to do the following tasks:

- Extract MSB of s0
- Clear LSB of s0
- Set LSB of s0 (bits 7 to 0 are set to 1)
- Clear s0 (s0=0, must use logical instructions)

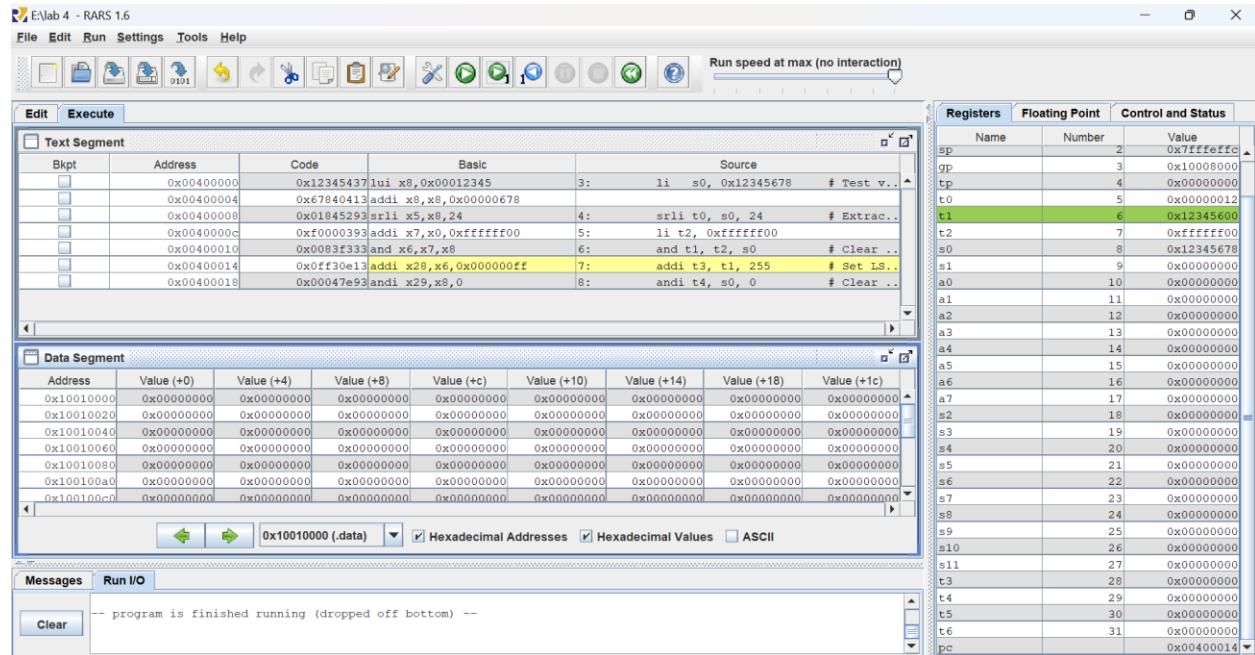
We initialize s0=0x12345678



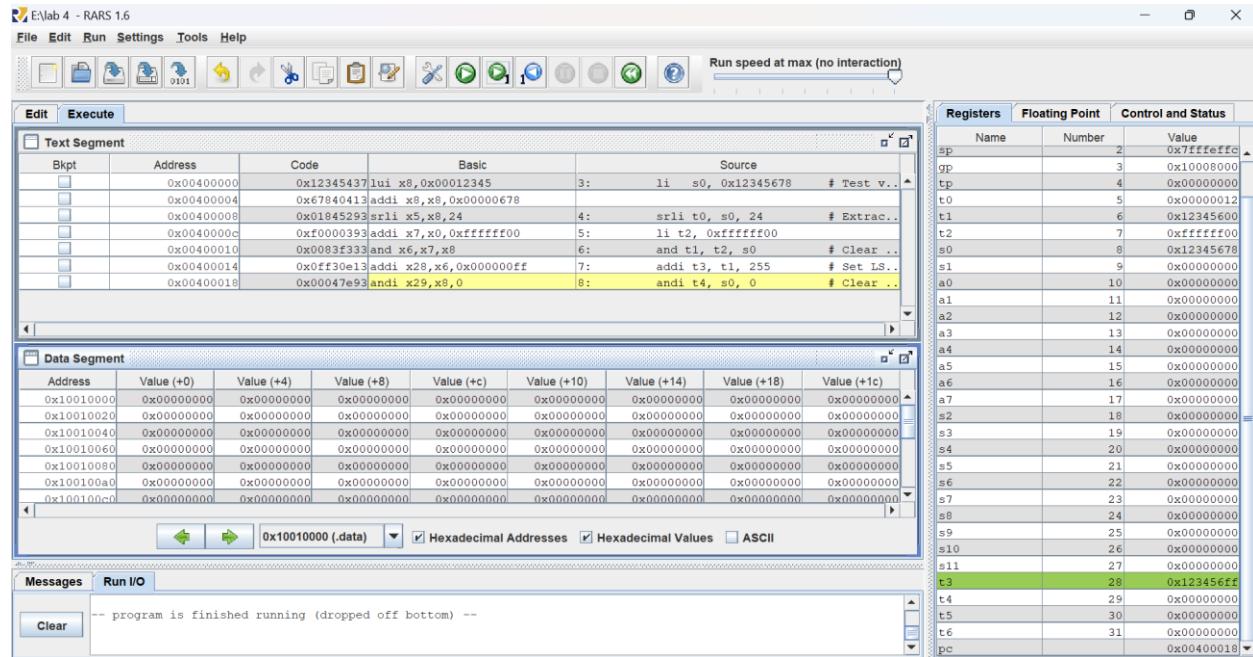
- Extract MSB of s0: t0=0x12



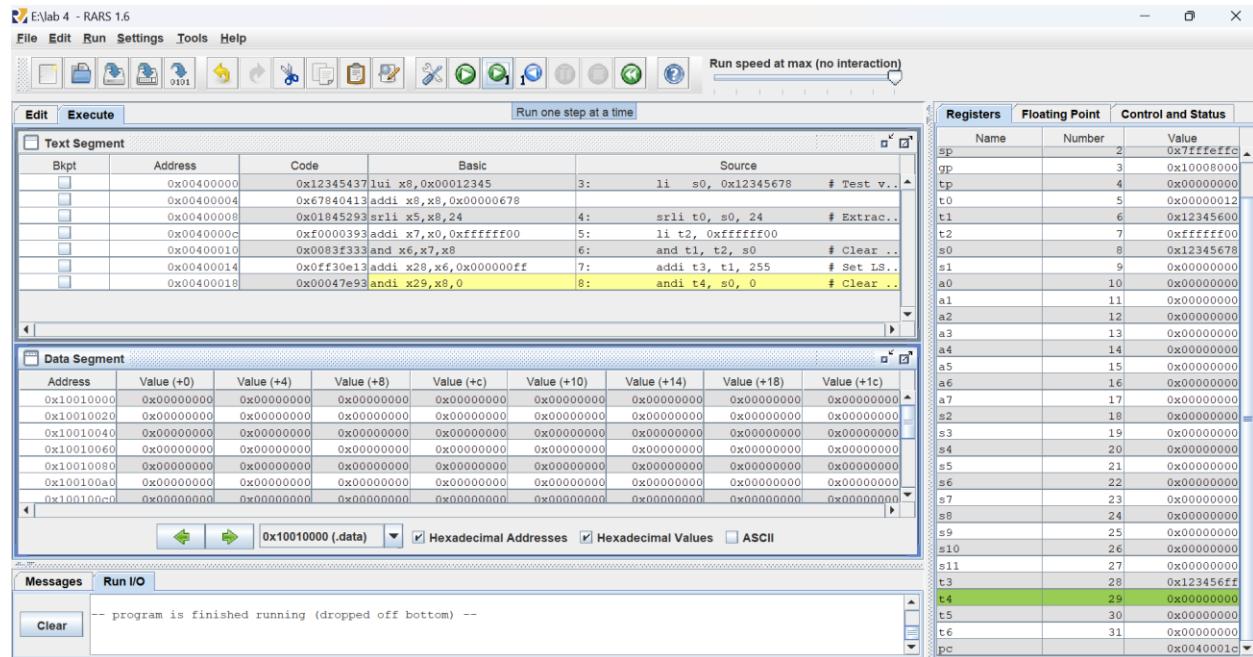
- Clear LSB of s0: t1=0x12345600



- Set LSB of s0 (bits 7 to 0 are set to 1) : t3=0x123456ff



- Clear s0 (s0=0, must use logical instructions): t4=0x00000000



Assignment 3: Pseudo instructions in RISC-V are not-directly-run-on-RISC-V-processor instructions which need to be converted to real-instructions of RISC-V. Re-write the following pseudo instructions using real-instructions understood by RISC-V processors:

a.

E\Lab 4 - RARS 1.6

Registers

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	45
s1	9	-45
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194312

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0xfd300493	addi x9,x0,-45	5: li s1, -45
	0x00400004	0x40900433	sub x8,x0,x9	6: sub s0, zero, s1

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

Messages

-- program is finished running (dropped off bottom) --

E\Lab 4 - RARS 1.6

Registers

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	45
s1	9	-45
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194312

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0xfd300493	addi x9,x0,-45	5: li s1, -45
	0x00400004	0x40900433	sub x8,x0,x9	6: sub s0, zero, s1

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

Messages

-- program is finished running (dropped off bottom) --

b.

E:\lab 4 - RARS 1.6

File Edit Run Settings Tools Help

Edit Execute

lab 4

```

1 # Laboratory Exercise 4, Assignment 3
2 .text
3     # Pseudo instructions
4     # move s0, s1: s0=s1
5     li s1, 15
6     add s0, zero, s1
7

```

Run speed at max (no interaction)

E:\lab 4 - RARS 1.6

File Edit Run Settings Tools Help

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00f00493addi x9,x0,15	5: li s1, 15	
	0x00400004	0x00900433add x8,x0,x9	6: add s0, zero, s1	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Registers

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	15
s1	9	15
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
gc		4194312

Messages Run I/O

-- program is finished running (dropped off bottom) --

Clear

C.

ELab 4 - RARS 1.6

Registers

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0xedcba987
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040000c

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x12345437	lui x8,0x00012345	5: li s0, 0x12345678
	0x00400004	0x67040413	addi x8,x8,0x00000678	
	0x00400008	0xffff44413	xori x8,x8,0xffffffff	6: not s0, s0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0xedcba987
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040000c

Messages

-- program is finished running (dropped off bottom) --

Registers

Name	Number	Value
sp	2	0x7ffffeffc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0xedcba987
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040000c

Messages

-- program is finished running (dropped off bottom) --

d.

The screenshot shows the ELab 4 - RARS 1.6 debugger interface. The assembly code in the editor window is:

```

1 # Laboratory Exercise 4, Assignment 3
2 .text
3     # Pseudo instructions
4     # ble s1, s2, label:
5     # if (s1 <= s2)
6     # j label
7     li s0, 0
8     li s1, 15
9     li s2, 23
10    ble s1, s2, YES
11    j EXIT
12 YES:
13    addi s0, zero, 1
14 EXIT:

```

The Registers window shows the following values:

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
t0	5	0
t1	6	0
t2	7	0
s0	8	1
s1	9	15
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	23
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194328

The Messages window shows the message: "program is finished running(dropped off bottom)".

S0=1, it means s1<=s2:

The screenshot shows the ELab 4 - RARS 1.6 debugger interface. The assembly code in the editor window is the same as the previous screenshot:

```

1 # Laboratory Exercise 4, Assignment 3
2 .text
3     # Pseudo instructions
4     # ble s1, s2, label:
5     # if (s1 <= s2)
6     # j label
7     li s0, 0
8     li s1, 15
9     li s2, 23
10    ble s1, s2, YES
11    j EXIT
12 YES:
13    addi s0, zero, 1
14 EXIT:

```

The Registers window shows the same values as the first screenshot.

The Text Segment window shows the following breakpoints:

Bkpt	Address	Code	Basic	Source
1	0x00400000	0x00000413	addi x8,x0,0	7: li s0, 0
2	0x00400004	0x00f00493	addi x9,x0,15	8: li s1, 15
3	0x00400008	0x01700913	addi x10,x0,23	9: li s2, 23
4	0x0040000c	0x00995463	bge x10,x9,8	10: ble s1, s2, YES
5	0x00400010	0x0080006f	jal x0,8	11: j EXIT
6	0x00400014	0x00100413	addi x8,x0,1	13: addi s0, zero, 1

The Data Segment window shows memory dump starting at address 0x10010000:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

The Messages window shows the message: "program is finished running(dropped off bottom)".

Assignment 4: To detect overflow in additional operation, we also use other rule than the one in Assignment 1. This rule is: when add two operands that have the same sign, overflow will occur if the sum doesn't have the same sign with either operands. You need to use this rule to write another overflow detection program.

The screenshot shows the RARS 1.6 assembly editor interface. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various icons for file operations and simulation. A status bar at the bottom indicates "Run speed at max (no interaction)". The main window has tabs for "Edit" and "Execute". The "Edit" tab is active, displaying the assembly code for "lab 4". The code implements a simple addition and overflow detection algorithm. The "Execute" tab is also visible. At the bottom left, there are buttons for assembly, binary, and hex dump, along with a search bar. The bottom right shows scroll bars.

```

1 # Laboratory Exercise 4, Assignment 4
2 .text
3     li $1, 1000000000
4     li $2, 2000000000
5     li $0, 0      # No overflow is default status
6     add $3, $1, $2      # $3 = $1 + $2
7     xor $1, $1, $2      # Test if $1 and $2 have the same sign
8     blt $1, zero, EXIT    # If not, exit
9     xor $2, $3, $1
10    bgt $2, zero, EXIT    # if $1, $3 have the same sign, exit
11    j OVERFLOW      # otherwise, overflow occurs
12
13 OVERFLOW:
14     li $0, 1      # The result is overflow
15 EXIT:

```

Line: 12 Column: 2 Show Line Numbers

We see that $t_0=1$, so overflow occurred in this program

The screenshot shows the RARS 1.6 debugger interface. The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various icons for file operations and simulation. A status bar at the bottom indicates "Run speed at max (no interaction)". The main window has tabs for "Edit" and "Execute". The "Execute" tab is active, showing the "Text Segment" and "Data Segment" panes. The "Text Segment" pane lists assembly instructions with their addresses and source code. The "Data Segment" pane shows memory values at specific addresses. On the right, a "Registers" pane displays the state of all 32 general-purpose registers (r0 to r31) and the PC. The bottom pane shows the "Messages" and "Run I/O" sections, with the message "program is finished running (dropped off bottom) --" displayed.

Name	Number	Value
sp	2	2147479548
gp	3	268468224
tp	4	0
r0	5	1
t1	6	1286561280
t2	7	-1991601152
s0	8	0
s1	9	1000000000
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	2000000000
s3	19	-1294967296
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194352

Assignment 5: Write a program that implement multiply by a small power of 2. (2, 4, 8, 16, etc for example).

```

1 # Laboratory Exercise 4, Assignment 5
2 .text
3 li $0, 5      # initialize the exponent
4 beq $0, zero, EXIT
5 li $t0, 1
6 li $s1, 1      # the result
7 li $t1, 2
8 loop:
9 bgt $t0, $0, EXIT
10 mul $s1, $s1, $t1
11 addi $t0, $t0, 1      #t0=t0+1
12 j loop
13 EXIT:

```

We have s_0 is the exponent and its value is up to us

We store the value of 2^{s_0} in the register s_1

In the example below, $s_0=5$. So, we get value $2^5=32$ in the register s_1 . In other words, $s_1=32$.

Name	Number	Value
\$sp	2	21474679548
gp	3	268466224
tp	4	0
t0	5	6
t1	6	2
t2	7	0
s0	8	5
s1	9	32
a0	10	0
a1	11	0
a2	12	0
a3	13	0
a4	14	0
a5	15	0
a6	16	0
a7	17	0
s2	18	0
s3	19	0
s4	20	0
s5	21	0
s6	22	0
s7	23	0
s8	24	0
s9	25	0
s10	26	0
s11	27	0
t3	28	0
t4	29	0
t5	30	0
t6	31	0
pc		4194344